Sesión 3: Secciones Críticas, Exclusión Mutua y Espera Activa

Concurrencia

Ángel Herranz

Febrero 2019

Universidad Politécnica de Madrid

Concurrencia

👍 Simultaneidad

+

Sincronización + 👍 Comunicación

Concurrencia

👍 Simultaneidad

+

Sincronización + 👍 Comunicación

Recordad

```
public static volatile int x = 0;
```

```
public class Inc {
  extends Thread
  public void run() {
    x = x + 1;
  }
}
```

```
public class Dec
  extends Thread {
  public void run() {
    x = x - 1;
  }
}
```

Recordad

```
public static volatile int x = 0;
```

```
public class Inc {
  extends Thread
  public void run() {
    x = x + 1;
  }
}
```

```
public class Dec
  extends Thread {
  public void run() {
    x = x - 1;
  }
}
```

Secciones Críticas

Concepto: condición de carrera

Resultados indeseados por interacción de dos o más procesos

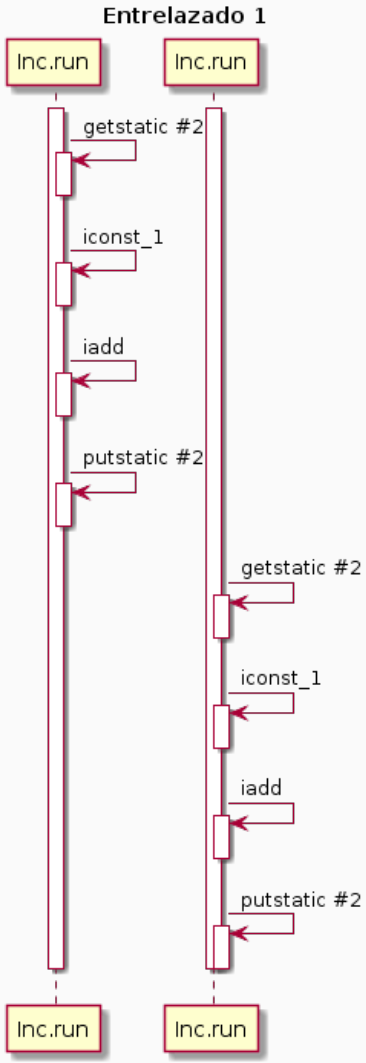
=

condición de carrera

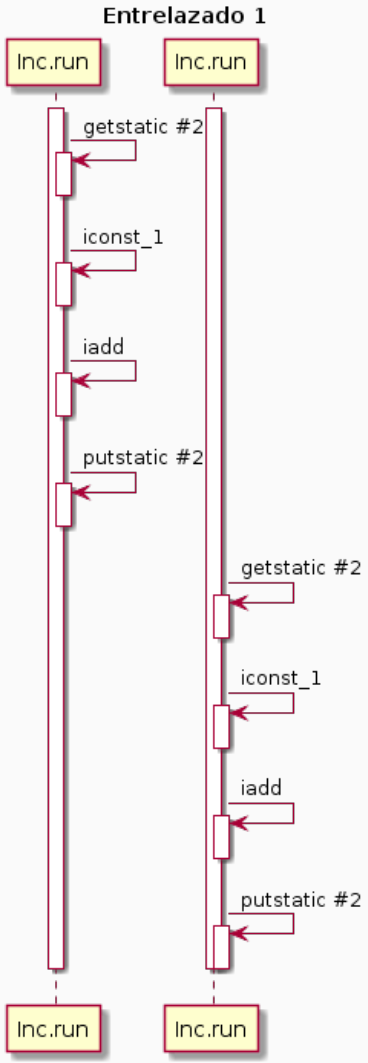
¿Por qué? Acciones atómicas

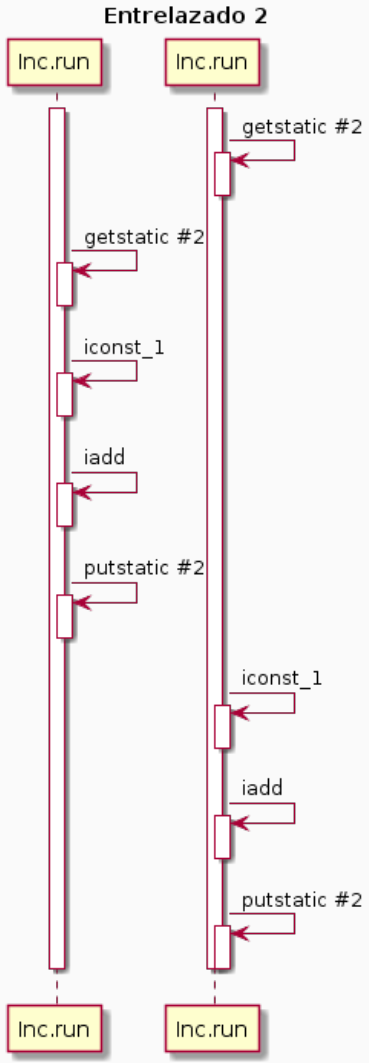
```
$ javap -c Inc
public class Inc extends Thread {
  ...
  public void run();
    Code:
       0: getstatic     #2                  // Field x:I
       3: iconst_1
       4: iadd
       5: putstatic     #2                  // Field x:I
       8: return
  ...
}
```

¿Por qué? Entrelazado

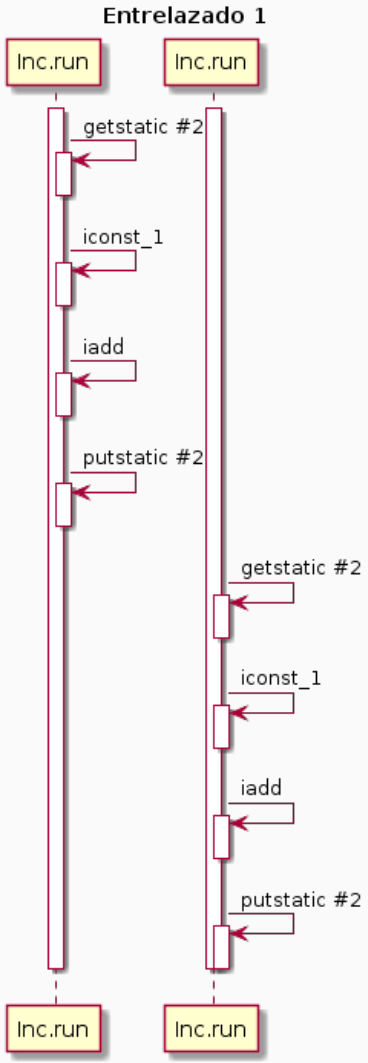


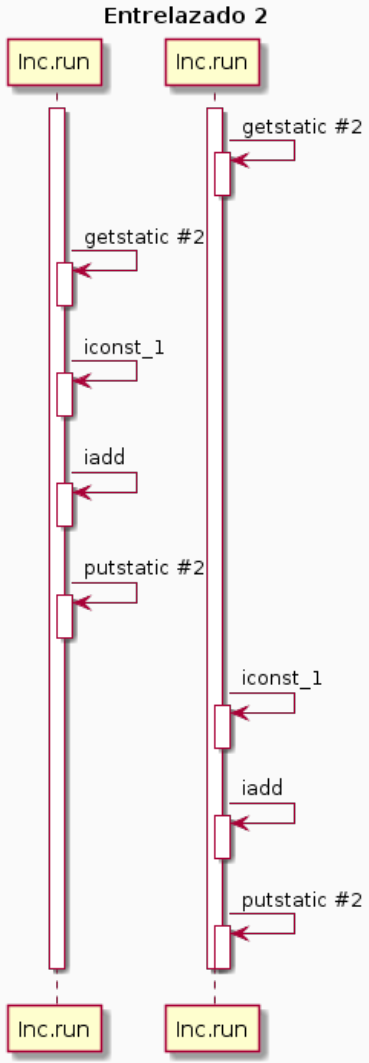
¿Por qué? Entrelazado

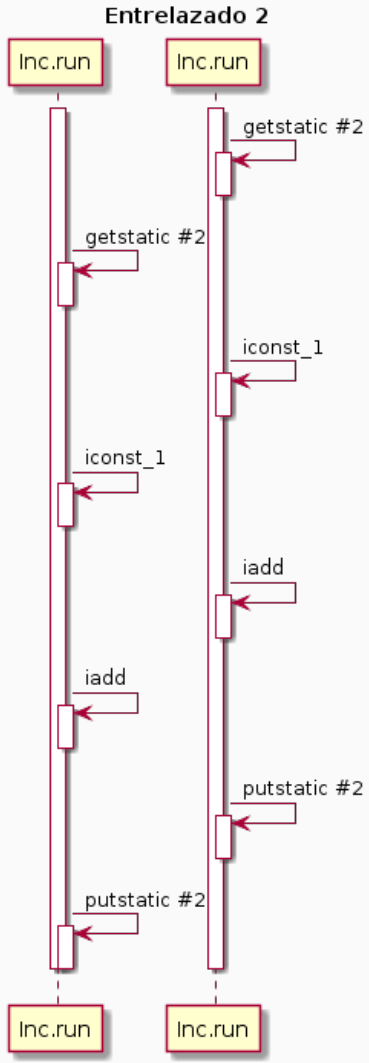




¿Por qué? Entrelazado







Concepto: sección crítica

Porción de código que puede dar lugar a una condición de carrera

=

sección crítica

Solución: Sincronización

Mientras un proceso está ejecutando la sección crítica, los otros procesos deben esperar

¿Cómo esperar?

```
// Esperar hasta
// que se cumpla C
while (!C) {}
```

¿Cómo esperar?

```
// Esperar hasta
// que se cumpla C
while (!C) {}
```

Espera Activa

Primer intento

```
public static volatile int x = 0;
```

```
public void run() {
  x = x + 1;
  }
}
```

```
public void run() {
  x = x - 1;
  }
}
```

Primer intento

```
public static volatile int x = 0;
public static volatile boolean enSC = false;

public void run() {
  x = x + 1;
  }
}
```

```
public void run() {
  x = x - 1;
  }
}
```

Primer intento

```
public static volatile int x = 0;
public static volatile boolean enSC = false;

public void run() {
  enSC = true;
  x = x + 1;
  enSC = false;
}
}
```

```
public void run() {
  enSC = true;
  x = x - 1;
  enSC = false;
}
}
```

Primer intento

```
public static volatile int x = 0;
public static volatile boolean enSC = false;

public void run() {
  while (enSC) {}
  enSC = true;
  x = x + 1;
  enSC = false;
}
}
```

```
public void run() {
  while (enSC) {}
  enSC = true;
  x = x - 1;
  enSC = false;
}
}
```